# On the Parameterized Complexity of Clique Elimination Distance

**Akanksha Agrawal**
Indian Institute of Technology Madras, Chennai, India
akanksha@iitm.ac.in

**M. S. Ramanujan**
University of Warwick, Coventry, UK
R.Maadapuzhi-Sridharan@warwick.ac.uk

## Abstract

Bulian and Dawar [Algorithmica, 2016] introduced the notion of elimination distance in an effort to define new tractable parameterizations for graph problems and showed that deciding whether a given graph has elimination distance at most $k$ to any minor-closed class of graphs is fixed-parameter tractable parameterized by $k$ [Algorithmica, 2017].

In this paper, we consider the problem of computing the elimination distance of a given graph to the class of cluster graphs and initiate the study of the parameterized complexity of a more general version – that of obtaining a modulator to such graphs. That is, we study the $(\eta, \textsc{Clq})$-Elimination Deletion problem ($(\eta, \textsc{Clq})$-ED Deletion) where, for a fixed $\eta$, one is given a graph $G$ and $k \in \mathbb{N}$ and the objective is to determine whether there is a set $S \subseteq V(G)$ such that the graph $G - S$ has elimination distance at most $\eta$ to the class of cluster graphs.

Our main result is a polynomial kernelization (parameterized by $k$) for this problem. As components in the proof of our main result, we develop a $k^{\mathcal{O}(\eta k+\eta^2)} n^{\mathcal{O}(1)}$-time fixed-parameter algorithm for $(\eta, \textsc{Clq})$-ED Deletion and a polynomial-time factor-$\mathsf{min}\{\mathcal{O}(\eta \cdot \mathsf{opt} \cdot \log^2 n), \mathsf{opt}^{\mathcal{O}(1)}\}$ approximation algorithm for the same problem.

**2012 ACM Subject Classification** Theory of computation → Fixed parameter tractability

**Keywords and phrases** Elimination Distance, Cluster Graphs, Kernelization

**Digital Object Identifier** 10.4230/LIPIcs.IPEC.2020.1

**Funding** *Akanksha Agrawal*: supported by the PBC Program of Fellowships for Outstanding Post-doctoral Researchers from China and India when a part of the work was carried out.

## 1 Introduction

A popular methodology for studying the parameterized complexity of problems is to consider *parameterization by distance from triviality* [19]. In this methodology, the idea is to try and lift the tractability of special cases of generally hard computational problems, to tractability of instances that are "close" to these special cases (i.e., close to triviality) for appropriate notions of "distance from triviality". With some exceptions (see, for example, [13, 21]), this approach typically has two components – (i) recognition algorithms that determine whether the input is indeed close to triviality and possibly compute a (approximate) witness, and (ii) solution algorithms that exploit the structure expressed by the witness in order to solve computational problems. In graph problems, a standard measure of distance from triviality is the size of a vertex modulator to a specific graph class, i.e., a set of vertices whose deletion results in a graph belonging to a specific graph class. This way of parameterizing graph problems has led to a rich collection of sophisticated algorithmic and lower bound machinery over the last two decades. Of particular interest to us in this work are two specific strands of research that fall under this framework.

In one strand, the goal is to enhance existing notions of distance from triviality by exploiting some form of *structure* underlying vertex modulators rather than just the size bound. This line of exploration has led to the development of several new notions of distance from triviality [11, 5, 6, 17, 16, 10]. Of special interest to us in this line of research is the notion of *elimination distance* introduced in [5]. Bulian and Dawar [5] introduced the notion of elimination distance in an effort to define tractable parameterizations that are more general than the modulator size for graph problems. We refer the reader to Section 2 for a formal definition of this parameter. In their work, they focused on the Graph Isomorphism (GI) problem and showed that GI is fixed-parameter tractable (FPT) when parameterized by the elimination distance to graphs of bounded degree. In follow-up work, Bulian and Dawar [6] showed that deciding whether a given graph has elimination distance at most $k$ to any minor-closed class of graphs is fixed-parameter tractable parameterized by $k$ (i.e., can be solved in time $f(k)n^{\mathcal{O}(1)}$). The second strand focuses on enhancing the set of "base classes" that capture the notion of triviality and study algorithms that compute or use small modulators to these classes. For instance, the computation and use of modulators into various hereditary graph classes (see, e.g., [14, 18, 3, 15, 24] for a partial list relevant to this paper) has been extensively explored.

In more recent years, efforts have been made to simultaneously build upon these two strands by retaining the notion of small modulators as the measure of distance from triviality, but enhancing the notion of triviality to include graphs that have bounded elimination to a second, well-understood graph class. Hols et al. [20] recently presented a comprehensive study of the classic Vertex Cover problem parameterized by the size of a smallest modulator to graphs that have bounded elimination distance to specific hereditary graph classes. They provided an elegant (partial) characterization of parameterizations that permit polynomial kernelizations for Vertex Cover. However, their focus is on solution (utilising the modulator) rather than recognition, and so they do not focus on computing the modulators.

In our current work, we draw from both strands of research described above and study the parameterized complexity of the $(\eta, \textsc{Clq})$-Elimination Deletion problem, where one is given a graph $G$ and $k \in \mathbb{N}$ with the objective of determining whether there is a set $S \subseteq V(G)$ such that the graph $G - S$ has elimination distance at most $\eta$ to the class of cluster graphs (disjoint union of cliques). We call graphs with elimination distance at most $\eta$ to the class of cluster graphs, $(\eta, \mathsf{Clq})$-graphs. Our parameter is the size of the modulator $k$ and we note that even for $k = 0$ (i.e., deciding whether the given graph has elimination distance at most $\eta$ to cluster graphs), this problem is quite non-trivial and even an algorithm with running time $n^{f(\eta)}$ is far from obvious. Indeed, Bulian and Dawar [6] ask about the possibility of extending their approach to obtain a fixed-parameter algorithm for this very problem.

The following is the formal definition of our main problem.

$(\eta, \textsc{Clq})$-Elimination Deletion($(\eta, \textsc{Clq})$-ED Deletion)

*Input:* A graph $G$ and an integer $k$.
*Parameter:* $k$.
Question: Is there a set $S \subseteq V(G)$ of size at most $k$, such that $G - S$ is an $(\eta, \mathsf{Clq})$-graph?

The central result of this paper is Theorem 1.

▶ **Theorem 1.** $(\eta, \textsc{Clq})$-Elimination Deletion *admits a kernelization algorithm running in time $\eta^{O(\eta^2)} \cdot n^{\mathcal{O}(1)}$, that outputs an equivalent instance with $2^{\mathcal{O}(\eta)} \cdot k^{\mathcal{O}(1)}$ vertices, where $n$ is the number of vertices in the input graph.*

Theorem 1 is in fact a *polynomial kernelization* for $(\eta, \textsc{Clq})$-ED Deletion as $\eta$ is a constant that is part of the problem description. We have explicitly stated the dependence on $\eta$ in all our results.

A simple corollary of Theorem 1 (obtained by setting $k = 0$) is an algorithm with running time $\eta^{\mathcal{O}(\eta^2)} \cdot n^{\mathcal{O}(1)}$ that determines whether a given graph is an $(\eta, \mathsf{Clq})$-graph (equivalently, we say that the $\mathsf{Clq}$-elimination distance of the given graph is at most $\eta$). That is, it is a fixed-parameter algorithm for recognising $(\eta, \mathsf{Clq})$-graphs parameterized by $\eta$. Towards the proof of Theorem 1, we prove the following two results of independent interest. The first of these results gives an approximation algorithm for $(\eta, \textsc{Clq})$-ED Deletion. The second result gives a moderately exponential-time fixed-parameter algorithm for $(\eta, \textsc{Clq})$-ED Deletion.

▶ **Theorem 2.** *There is an algorithm that, given a graph $G$ on $n$ vertices, runs in time $\eta^{O(\eta^2)} \cdot n^{\mathcal{O}(1)}$ and outputs a set $S \subseteq V(G)$ of size $\mathcal{O}(\eta \cdot \mathsf{opt}^2 \cdot \log^2 n)$, such that $G - S$ has $\mathsf{Clq}$-elimination distance at most $\eta$.*

▶ **Theorem 3.** $(\eta, \textsc{Clq})$-Elimination Deletion *can be solved in time $(k+\eta)^{O(\eta k+\eta^2)} n^{\mathcal{O}(1)}$.*

An important consequence of Theorem 2 and Theorem 3 is a polynomial-time $\mathsf{opt}^{\mathcal{O}(1)}$-approximation algorithm for $(\eta, \textsc{Clq})$-ED Deletion. We remark that the moderately exponential dependence on $k$ in the running time of Theorem 3 is crucially used in obtaining this approximation algorithm and hence, we avoid resorting to meta-theorems in the proof of Theorem 3. This $\mathsf{opt}^{\mathcal{O}(1)}$-approximation algorithm for $(\eta, \textsc{Clq})$-ED Deletion is the starting point of our proof of Theorem 1. Such $\mathsf{opt}^{\mathcal{O}(1)}$-approximation algorithms play a crucial role in bootstrapping kernelization algorithms (see, for example, [22, 2]). They also allow for further consequences when studying kernelization of problems parameterized by the size of the smallest modulator to $(\eta, \mathsf{Clq})$-graphs since we do not need to assume that the modulator is given as part of the input. In the literature on such structural parameterizations, it is generally assumed that the modulator is included in the input. Often, such an assumption can be made without loss of generality because the modulator can be $\mathsf{opt}^{\mathcal{O}(1)}$-approximated in polynomial time. However, there are situations where the assumption is necessary due to the lack of such approximations. We refer the reader to [12] for a detailed discussion on formalizing structural parameterizations.

Finally, as part of the proof of Theorem 3, we obtain a constant factor fixed-parameter approximation algorithm for the problem of determining the $\mathsf{Clq}$-elimination distance of a given graph. Moreover, we show that by invoking a result of Czerwiński et al. [8], one can speed up this algorithm at the cost of a worse approximation. The formal statement is given below (we refer the reader to Section 2 for the definition of an $(\eta, \mathsf{Clq})$-decomposition).

▶ **Theorem 4.** *There are algorithms $\mathcal{A}_1$, $\mathcal{A}_2$ such that, given a graph $G$ and an integer $\eta$, the following hold:*

1. *$\mathcal{A}_1$ runs in time $2^{\mathcal{O}(\eta^2)} \cdot n^{\mathcal{O}(1)}$ and either correctly reports that the $\mathsf{Clq}$-elimination distance of $G$ is more than $\eta$, or computes a $(5\eta, \mathsf{Clq})$-decomposition of $G$.*
2. *$\mathcal{A}_2$ runs in time $2^{\mathcal{O}(\eta)} \cdot n^{\mathcal{O}(1)}$ and either correctly reports that the $\mathsf{Clq}$-elimination distance of $G$ is more than $\eta$, or computes an $(\mathcal{O}(\eta^2 \log^{3/2} \eta), \mathsf{Clq})$-decomposition of $G$.*

## Related work

Recently, Lindermayr et al. [27] showed that computing elimination distance to bounded degree graphs is fixed-parameter tractable when the input is planar. Bougeret et al. [2] introduced a measure called bridge-depth and showed that a minor-closed family of graphs

$\mathcal{F}$ has bounded bridge-depth precisely when Vertex Cover admits a polynomial kernel parameterized by the size of a modulator to $\mathcal{F}$ (subject to standard complexity theoretic hypotheses). The notion of elimination distance [5] generalizes the notion of *generalized treedepth* introduced by Bouland et al [4] in an effort to combine the treedepth and max-leaf number parameters. Building on [5] and extending the approach of combining width parameters (treedepth in the case of [5]) and modulator size, Ganian et al. [17] proposed a measure of distance to triviality for CSP that depended on the treewidth of an appropriate graph defined on backdoor sets (these can be thought of as a version of vertex modulators appropriate for use in solving ILP and CSP instances). That is, they introduced a way of combining *treewidth* and modulator size into a single parameter that is stronger than elimination distance. More recently, Eiben et al. [10] continued the line of research into combining modulators and width parameters by studying this parameter in the context of graph problems, where triviality is expressed in terms of bounded rankwidth.

## 2 Preliminaries

We refer to the book of Diestel [9] for standard graph terminology. Whenever the context is clear, we use $n$ and $m$ to denote the number of vertices and the number of edges in the input graph, respectively. A *vertex cover* in $G$ is a set $S \subseteq V(G)$, such that $G - S$ has no edges. By $\mathsf{vc}(G)$ we denote the size of a minimum sized vertex cover in $G$. We say that a set $S \subseteq V(G)$ is a *clique* in $G$ if for every distinct $u, v \in S$, we have $\{u, v\} \in E(G)$. We let $\mathsf{Clq}(G)$ denote the set of all cliques in $G$.

▶ **Proposition 5** ([23, 28]). *We can generate all maximal cliques of a graph with $\mathcal{O}(n^{\omega})$ time delay, where $\omega$ is the exponent in the running time of matrix multiplication.*[1]

The set $\mathcal{C}(G)$ denotes the set of connected components of $G$. Consider a graph $G$. For sets $X, Y \subseteq V(G)$, an *$X$-$Y$ separator* in $G$ is a set $S \subseteq V(G)$, such that $G - S$ has no $x$-$y$ path, where $x \in X \setminus S$ and $y \in Y \setminus S$. By $\mathsf{sep}_G(X, Y)$ we denote the size of a minimum sized $X$-$Y$ separator in $G$. Our algorithm(s) will rely on existence of balanced separators, which is defined next.

▶ **Definition 6.** For a graph $H$, a set $Z \subseteq V(H)$ is a *balanced separator* of $H$, if the connected components of $H - Z$ can be partitioned into two sets, $\mathcal{C}_1$ and $\mathcal{C}_2$, such that $|\cup_{C \in \mathcal{C}_1} V(C)| \leq 2|V(H)|/3$ and $|\cup_{C \in \mathcal{C}_2} V(C)| \leq 2|V(H)|/3$.

For a tree $T$ and vertices $u, v \in V(T)$, we denote the unique path between $u$ and $v$ by $\mathsf{Pth}_T(u, v)$. A *rooted tree* is a tree with a special vertex called the *root* of the tree. Consider a rooted tree $T$ with root $r$. A vertex $t \in V(T) \setminus \{r\}$ is a *leaf* of $T$ if it is a vertex of degree exactly one in $T$. Moreover, if $V(T) = \{r\}$, then $r$ is the leaf (as well as the root) of $T$. A vertex which is not a leaf, is a *non-leaf* vertex. Consider a node $t \in V(T)$. We say that $t'$ is a *child* of $t$, if $\{t, t'\} \in E(T)$ and $t'$ does not belong to the unique $t - r$ path in $T$. Furthermore, we say that $t = \mathsf{par}_T(t')$ is the *parent* of $t'$. A vertex $t' \in V(T)$ ($t'$ can possibly be the same as $t$) is a *descendant* of $t$, if in $T - \{\mathsf{par}_T(t)\}$, where $\mathsf{par}_T(t)$ is the parent of $t$, there is a $t - t'$ path. Note that when $t = r$, then $T - \{\mathsf{par}_T(t)\} = T$, as the parent of $r$ does not exist. (Every vertex in $T$ is a descendant of $r$.) By $\mathsf{desc}_T(t)$, we denote the set of all descendants of $t$ in $T$. We drop the subscript $T$ from $\mathsf{par}_T(\cdot)$ and $\mathsf{desc}_T(\cdot)$, when the context is clear. A *rooted forest* is a forest where each of its connected component is a rooted tree. The set of

[1] The current best value of $\omega$ is less than 2.3727 [31].

leaves of a rooted forest is the union of the sets of leaves of the trees in this forest. The set of leaves in a rooted forest $F$ is denoted by $\mathsf{Lf}(F)$. The *depth*, denoted by $\mathsf{depth}(T)$ of a rooted tree $T$ is the maximum number vertices in a root to leaf path in $T$. The *depth*, denoted by $\mathsf{depth}(F)$ of a rooted forest is the maximum over the depths of its rooted trees. We now define the notion of tree decompositions.

▶ **Definition 7.** A *tree decomposition* of a graph $G$ is a pair $(T, \beta)$, where $T$ is a tree rooted at $r$ and $\beta : V(T) \to 2^{V(G)}$ that satisfies the following properties:
1. $\bigcup_{t \in V(T)} \beta(t) = V(G)$,
2. for every edge $\{u, v\} \in E(G)$ there is a node $t \in V(T)$, such that $u, v \in \beta(t)$, and
3. for every $v \in V(G)$, the graph $T[X_v]$ is a subtree of $T$, where $X_v = \{t \in V(T) \mid v \in \beta(t)\}$.

For $t \in V(T)$, we call $\beta(t)$ the *bag* of $t$. The sets in $\{\beta(t) \mid t \in V(T)\}$ are called *bags* of $(T, \beta)$. We refer to the vertices in $V(T)$ as *nodes*, to distinguish it from the vertices of $G$. The *width* of the tree decomposition $(T, \beta)$ is $\max_{t \in V(T)} |\beta(t)| - 1$. The *treewidth* of $G$, denoted by $\mathsf{tw}(G)$, is the minimum over the widths over all possible tree decompositions of $G$.

A tree decomposition $(T, \beta)$ of a graph $G$ is called a *path decomposition* if $T$ is a path. Moreover, *pathwidth* of $G$, denoted by $\mathsf{pw}(G)$, is the minimum over the widths over all possible path decompositions of $G$. In the following we state some folklore properties about bounded treewidth graphs, that will be useful later.

▶ **Proposition 8** (Exercise 7.6 [7]). *Consider a graph $G$, a tree decomposition $(T, \beta)$ of $G$, and a clique $S \subseteq V(G)$ in $G$. Then, there is $t \in V(T)$, such that $S \subseteq \beta(t)$.*

▶ **Proposition 9.** *For a graph $G$, the number of distinct cliques (not necessarily maximal) in $G$ is bounded by $\mathcal{O}(2^{\mathsf{tw}(G)} \cdot n)$.*

▶ **Definition 10** ([25, 7]). For a graph $H$, a path decomposition $\mathcal{P} = (P = (p_1, p_2, \cdots, p_t), \beta : V(P) \to 2^{V(H)})$ of $H$ is a *nice path decomposition* if $\beta(p_1) = \beta(p_t) = \emptyset$, $P$ is rooted at $p_t$, and every node $p_i$, for $i \in [t] \setminus \{1, t\}$ is of exactly one of the following types:
1. **Insert Vertex Node.** We have $\beta(p_i) = \beta(p_{i-1}) \cup \{v\}$, for some $v \in V(H) \setminus \beta(p_{i-1})$.
2. **Forget Vertex Node.** We have $\beta(p_i) = \beta(p_{i-1}) \setminus \{v\}$, for some $v \in \beta(p_{i-1})$.

We now state a result regarding computation of a nice path decomposition of a graph, which follows from Lemma 7.2 of [7] and Proposition 8.

▶ **Proposition 11.** *Any graph $H$ that admits a path decomposition of width at most $p$, also admits a nice path decomposition of width at most $p$. Moreover, given a path decomposition $\mathcal{P} = (P = (p_1, p_2, \cdots, p_t), \beta)$ of $H$ of width at most $p$, one can compute a nice path decomposition $\mathcal{P}' = (P' = (p'_1, p'_2, \cdots, p'_{t'}), \beta')$ of $H$ of width at most $p$ in time $\mathcal{O}(p^2 \cdot \max(|V(P)|, |V(H)|))$, such that the root node, $p_t$, is a forget node and for each $i \in [t]$, there is some $j \in [t']$, with $\beta(p_i) = \beta'(p'_j)$.*

▶ **Definition 12** (Forest embedding). A *forest embedding* of a graph $G$ is a pair $(F, f)$, where $F$ is a rooted forest and $f : V(G) \to V(F)$ is a bijective function, such that for each $\{u, v\} \in E(G)$, either $f(u)$ is a descendant of $f(v)$, or $f(v)$ is a descendant of $f(u)$. The *depth* of the forest embedding $(F, f)$ is the depth of the rooted forest $F$.[2] The *treedepth* of $G$, denoted by $\mathsf{td}(G)$, is the minimum over the depths over all possible forest embeddings of $G$.

[2] Sometimes we slightly abuse the notation for simplicity, and say that, for every $\beta \geq \alpha$, $(F, f)$ is a forest embedding of depth $\beta$, where $\alpha$ is the depth of $F$.

▶ **Definition 13** (Induced forest embedding). *Let $(F, f)$ be a forest embedding of $G$ and let $W \subseteq V(G)$. Define the pair $(F', f')$ as follows:*

- *$V(F') = V(F) \cap f(W)$ and $f' : W \to V(F')$ is defined as $f|_W$, where $f(W) = \bigcup_{w \in W} f(w)$.*
- *For every $u \in V(F')$, if no ancestor of $u$ in $F$ is in $f(W)$, then make $u$ a root.*
- *For every $u, v \in V(F')$ such that $u$ is an ancestor of $v$ in the rooted forest $F$, we add an edge between $u$ and $v$ (making $v$ a child of $u$ in $F'$) if and only if $W$ is disjoint from $f^{-1}(X)$, where $X$ is the set of internal vertices on the unique $u$-$v$ path in $F$.*

*We say that $(F', f')$ is the* forest embedding induced *by $(F, f)$* on the set *$W$.*

In the following we state some known results regarding treedepth of a graph.

▶ **Proposition 14** (see Excercise 7.54 [7]). *For every graph $G$, $\mathsf{tw}(G) \le \mathsf{td}(G)$.*

The next observation states that for every clique $S$ in $G$, and every forest embedding of $G$, there is a root-to-leaf path in this forest embedding that contains all the vertices of $S$.

▶ **Observation 15.** *Consider a graph $G$, a forest embedding $f : V(G) \to V(F)$ of $G$ into the rooted forest $F$, and a clique $S \subseteq V(G)$ in $G$. Then, there is a rooted tree $T \in \mathcal{C}(F)$*[3] *with root $r$ and a leaf $t \in V(T)$, such that for each $s \in S$, we have $f(s) \in V(\mathsf{Pth}_T(r, t))$.*

Next, we recall the notion of *elimination-distance* introduced by Bulian and Dawar [5]. We rephrase their definition and introduce notation that will facilitate our presentation.

▶ **Definition 16** (Elimination Distance and $(\eta, \mathcal{H})$-decompositions). Consider a family $\mathcal{H}$ of graphs and an integer $\eta \in \mathbb{N}$. An $(\eta, \mathcal{H})$-decomposition of a graph $G$ is a tuple $(X, Y, F, f : X \to V(F), g : \mathcal{C}(G[Y]) \to \mathsf{Lf}(F) \cup \{\perp\})$, where $(X, Y)$ is a partition of $V(G)$ and $F$ is a rooted forest of depth $\eta$, such that the following conditions are satisfied:

1. $(F, f)$ is a forest embedding of $G[X]$,
2. each connected component of $G[Y]$ belongs to $\mathcal{H}$, and
3. for a connected component $C$ of $G[Y]$, a vertex $v \in V(C)$, and an edge $\{u, v\} \in E(G)$, either $u \in Y$ or $f(u)$ is a vertex in the unique path in $F$ from $r$ to $g(C)$, where $r$ is the root of the connected component in $F$ containing the vertex $g(C)$.[4]

We say that $G$ admits an $(\eta', \mathcal{H})$-decomposition if there is some $\eta \le \eta'$, for which there is an $(\eta, \mathcal{H})$-decomposition of $G$. The *elimination distance* of $G$ to $\mathcal{H}$ (or the *$\mathcal{H}$-elimination distance* of $G$) is the smallest integer $\eta^*$ for which $G$ admits an $(\eta^*, \mathcal{H})$-decomposition.

Consider an $(\eta, \mathcal{H})$-decomposition $\mathbb{D} = (X, Y, F, f, g)$ of a graph $G$. We say that $X$ is the *interior part* of $\mathbb{D}$ and $Y$ is the *exterior part* of $\mathbb{D}$. For a leaf $u \in \mathsf{Lf}(F)$, by $\widehat{P}_u^{\mathbb{D}}$ we denote the the path from $u$ to $r$ in the tree $T$, where $T$ is the tree rooted at $r$ in $F$, containing $u$. Moreover, by $P_u^{\mathbb{D}}$, we denote the graph $G[\{f^{-1}(w) \mid w \in V(\widehat{P}_u^{\mathbb{D}})\}]$. For a connected component $C \in \mathcal{C}(G[Y])$, by $C_{\mathsf{ext}}^{\mathbb{D}}$ we denote the graph $G[V(C) \cup \{f^{-1}(w) \mid w \in V(\mathsf{Pth}_F(g(C), r))\}]$, where $r$ is the root of the component of $F$ containing $g(C)$. (For the above notations we drop the superscript $\mathbb{D}$, when the context is clear.) The following observation directly follows from the definition of $C_{\mathsf{ext}}^{\mathbb{D}}$ and item 3 of Definition 16.

▶ **Observation 17.** *Consider a graph $G$ with an $(\eta, \mathsf{Clq})$-decomposition $\mathbb{D} = (X, Y, F, f, g)$. For every clique $S$ in $G$ the following holds: i) if $S \cap Y = \emptyset$, then there is $u \in \mathsf{Lf}(F)$, such that $S \subseteq V(P_u)$, otherwise, ii) $S \cap Y \neq \emptyset$, and there is $C \in \mathcal{C}(G[Y])$, such that $S \subseteq V(C_{\mathsf{ext}})$.*

[3] Recall that $\mathcal{C}(F)$ denotes the set of connected components of $F$,
[4] If $g(C) = \perp$, then $u$ must belong to $Y$.

For a graph $G$ and integer $\eta$, by $\mathsf{opt}_\eta(G)$, we denote the size of a minimum sized set $S \subseteq V(G)$, such that $G - S$ admits an $(\eta, \mathsf{Clq})$-decomposition. We drop the argument $G$ and the subscript $\eta$, whenever the context is clear. For a set $S \subseteq V(G)$, we say that $S$ is *solution*, if $G - S$ has $\mathsf{Clq}$-elimination distance at most $\eta$. Furthermore, we say that $S$ is a $t$-solution if $|S| \leq t$.

## 3 Overview of our algorithms

In this section, we give high level summaries of the proofs behind our results. For all our algorithms, we assume that the input graph is connected and not a clique.

### 3.1 Polynomial kernelization for $(\eta, \mathsf{Clq})$-ED Deletion (Theorem 1)

We first outline our polynomial kernelization assuming Theorem 2 and Theorem 3.

Our kernelization is heavily inspired by the work of Fomin et al. [14] and Giannopoulou et al. [18]. Fomin et al. gave a polynomial kernelization for the $\eta$-Treewidth Deletion problem (and more generally, for the Planar $\mathcal{F}$-deletion problem). Their kernel has size $k^{f(\eta)}$ and subsequently, Giannopoulou et al. [18] developed specialised reduction rules that result in a kernel of size $f(\eta) \cdot k^6$ for the special case of $\eta$-Treedepth Deletion. In both these kernelizations, the starting point is a constant-factor approximate modulator to $\eta$-treewidth graphs (respectively, $\eta$-treedepth graphs). While an approximate $\eta$-treedepth modulator can be obtained by repeatedly taking the vertex set of a sufficiently long path in the graph into the modulator, such an approach will not help in our case and we rely on Theorem 2 and Theorem 3 to obtain a polynomial-time $\mathsf{opt}^{\mathcal{O}(1)}$-approximation algorithm for $(\eta$, Clq)-ED Deletion.

Indeed, suppose that $n \leq (k+\eta)^{d(\eta k+\eta^2)}$ ($d$ is the constant hidden in the $\mathcal{O}(\cdot)$ notation in Theorem 3). Then, Theorem 2 implies a $\mathsf{opt}^{\mathcal{O}(1)}$-approximation in polynomial time. On the other hand, if $n > (k+\eta)^{d(\eta k+\eta^2)}$, then the algorithm of Theorem 3 runs in polynomial time. This approximation algorithm is the starting point of our proof of Theorem 1.

We now describe the rest of our kernelization algorithm. Using Theorem 4, we obtain a $(5\eta+1, \mathsf{Clq})$-decomposition, $\widetilde{\mathbb{D}} = (X, Y, T, f, g)$, of $G - S$, where $T$ is a rooted tree. The two main objectives of the algorithm are to i) bound the size of a connected component of $G[Y]$ by $(k+\eta)^{\mathcal{O}(1)}$, and ii) bound the degree of a vertex in $T$ (and also the number of connected components of $G[Y]$ associated with a leaf in $T$ via the function $g$) by $2^{\mathcal{O}(\eta)} \cdot k^{\mathcal{O}(1)}$. Once we achieve the above two, using the fact that $T$ has bounded depth, we can obtain our kernel of the desired size. Each of our reduction rules either decreases the number of non-edges in the input, or deletes a vertex. Thus in total, our algorithm will apply at most $n^2 + n$ reduction rules. We will next give an intuitive description of our reduction rules (and their workings).

Our first reduction rule helps us bound the size of a connected component of $G[Y]$, and we briefly explain the working of this reduction rule, below. Consider a connected component $C$ of $G[Y]$, and let $a = g(C)$. Furthermore, let $X_a$ be the set of vertices from $X$, that are mapped to the vertices in the path from $a$ to the root of $T$. For each $u \in X_a \cup S$, we mark $\mathcal{O}(k+\eta)$ neighbors (and non-neighbors) of $u$ in $C$. After this, our reduction rules remove all unmarked vertices from $C$. The idea behind the correctness of the above reduction rule is that if $u$ has large neighborhood in $C$, then any solution $S^*$ to $G$ can delete at most $k$ of these neighbors. Moreover, at most $\eta$ of these neighbors of $u$ can belong to the interior part of the decomposition for $G - S^*$. Thus, if we mark $\mathcal{O}(k+\eta)$ neighbors of $u$ in $C$, then we will be able to preserve the information that some neighbors of $u$ in $C$ must belong to the exterior part of the decomposition. Once we have the above property, we may add the deleted vertex from $C$ to the exterior of $(\eta, \mathsf{Clq})$-decomposition for the reduced instance, to obtain such a decomposition for the original instance.

Towards designing our reduction rule for bounding the degree of vertices in $T$, we first devise another reduction rule that, very roughly speaking, helps up ensure that for the neighborhood of the (sub-)graph below a vertex $a \in V(T)$, in $S$, induces a clique. This clique-neighborhood property helps us to avoid considering the exact set of neighborhood in $S$, as the vertices from a clique must belong to a root to leaf path, in any $(\eta, \mathsf{Clq})$-decomposition. To achieve the clique-neighborhood property, we add edges between non-adjacent pair of vertices in $S$ that have $(k+\eta)^{\Omega(1)}$-flow between them (the flow value is proportional to bound on the size of a connected components in $G[Y]$). On the other hand, for pairs of vertices in $S$ that do not have $(k+\eta)^{\Omega(1)}$-flow between them, we mark the vertices of a minimum separator for these vertices contained in $G - S$. As the number of separator vertices thus marked is bounded by $(k+\eta)^{\mathcal{O}(1)}$, they do not contribute excessively to the degree of a vertex in $T$. Fix a vertex $a$ in $T$ that has an unbounded number of children and let $a_1, \ldots, a_\ell$ be the children of $a$ in $T$. For $i \in [\ell]$, let $G_i$ be the graph $G[V_i]$, where $V_i$ contains all i) $v \in X$, for which $f(v)$ is a descendant of $a_i$, and ii) all vertices of $C \in \mathcal{C}(G[Y])$, where $g(C)$ is a descendant of $a_i$. A child $a_i$ of $a$ is *relevant*, if $G_i$ does not contain a marked separator vertex. At this point, for a relevant child $a_i$ of $a$ in $T$, $N(V(G_i)) \cap S$ is a clique. Let $X_a \subseteq X$ be the set of vertices that are mapped to vertices in the path from $a$ to the root of $T$. For each $B \subseteq X_a$, we mark $\mathcal{O}(k+\eta)$ relevant children $a_i$ of $a$, for which $N(V(G_i)) \cap X_a$ is $B$. Also, for each $s \in S$, we mark $\mathcal{O}(k+\eta)$ relevant child $a_i$ of $a$, for which $V(G_i)$ has $s$ as a neighbor (resp. non-neighbor). While marking relevant children in this way, we not only consider their neighborhoods as described, but we also do the following: for every possible $\widehat{\eta} \in [\eta]_0$, mark $\mathcal{O}(k+\eta)$ relevant children $a_i$ (satisfying the aforementioned constraints on their neighborhood) that have elimination distance exactly $\widehat{\eta}$. Following this series of markings, if a child $a_i$ of $a$ is unmarked, then we delete all the vertices in $V(G_i)$, from $G$. Since we have preserved $\mathcal{O}(k+\eta)$ representatives for the neighborhood and the elimination distance, given a solution $\widehat{S}$ for $G - V(G_i)$, and an $(\eta, \mathsf{Clq})$-decomposition for $(G - V(G_i)) - S$, we will be able to show that $G_i$ can be appended in an identical fashion to one of preserved representatives, thus giving an $(\eta, \mathsf{Clq})$-decomposition for $G - S$. Summing up, when no reduction rules are applicable, we can bound the number of vertices in the graph by $2^{\mathcal{O}(\eta)} \cdot k^{\mathcal{O}(1)}$.

## 3.2 FPT-algorithm for $(\eta, \mathsf{Clq})$-ED Deletion (Theorem 3)

We now give a summary of the proof of Theorem 3 assuming Theorem 4. The first ingredient of our FPT algorithm is the well-known technique of iterative compression, introduced by Reed, Smith, and Vetta [29]. Roughly speaking, using the above, we can reduce our goal to solving a variant of $(\eta, \textsc{Clq})$-Elimination Deletion, where we have a $(k+1)$-solution, call it $Z$, for $G$, and the objective is to find a $k$-solution, $S \subseteq V(G) \setminus Z$ for $G$. As $Z$ is a solution for $G$, $G - Z$ admits an $(\eta, \mathsf{Clq})$-decomposition. We compute a $(5\eta, \mathsf{Clq})$-decomposition, $\mathbb{D} = (X, Y, F, f, g)$, of $G - Z$ using Theorem 4. Using this $(5\eta, \mathsf{Clq})$-decomposition $\mathbb{D}$, we compute a path decomposition $\mathcal{P} = (P, \beta : V(P) \to 2^{X \cup Z})$ of $G[X \cup Z]$. We then compute a "pathlike decomposition", $\mathcal{T}$ of $G$, by attaching connected components of $G[Y]$ (which induce cliques), as bags, to $\mathcal{P}$. We remark that $\mathcal{T}$ may not be a tree-decomposition of $G$, as the edges between vertices in $Y$ and $X \cup Z$ may not be contained in any bag of $\mathcal{T}$. We ensure that each bag of $\mathcal{P}$ is attached to at most one connected component of $G[Y]$ (which is achieved by repeating some bags of $\mathcal{P}$, when necessary), and for any connected component $C$ of $G[Y]$, the neighborhood of $C$ is contained in the bag of $\mathcal{P}$, to which it is attached. We then execute a dynamic programming algorithm over this pathlike decomposition. Roughly speaking, we define states only for bags that are present in $\mathcal{P}$ and only preserve polynomial-size information regarding the cliques (from $G[Y]$), attached to it. Remembering only polynomial sized information regarding the attached cliques is possible due to the following two properties: i) at most one connected component of $G[Y]$ is attached to a bag from $\mathcal{P}$, which allows us to use "vertex cover" like property to identify vertices that must go to the interior part of the decomposition. ii) As vertices of the connected component $C$ of $G[Y]$ induces a clique, the vertices from $C$ that go to the interior, must belong to one root to leaf path.

We will now intuitively discuss the states for our dynamic programming algorithm. Let $P = (p_1, p_2 \cdots, p_\ell)$, where $p_\ell$ is the root of $P$. For $i \in [\ell]$, let $G_i$ be the graph induced on vertices that appear in $\beta(p_{i'})$ and the clique attached to it (if any), for $i' \in \{1, 2, \cdots, i\}$. Suppose that $S$ is a solution for $G$, that we are looking for, and $\mathbb{D}' = (X', Y', F', f' : V(X') \to V(F'), g' : \mathcal{C}(G[Y']) \to V(F') \cup \{\perp\})$ is an $(\eta, \mathsf{Clq})$-decomposition for $G - S$.[1] For each $i \in [\ell]$, we will maintain a guess for the partition, $(\widehat{X}, \widehat{Y}, \widehat{W})$ of $\beta(p_i)$, where we want $\widehat{W} = S \cap \beta(p_i)$, $\widehat{X} = X' \cap \beta(p_i)$, and $\widehat{Y} = Y' \cap \beta(p_i)$. We will also maintain the information regarding the "structure" of $(F', f', g')$, when restricted to the vertices in $\beta(p_i) - S$. Although we cannot maintain the whole of $F'$, we will maintain a tuple $\mathcal{F} = (\widehat{F}, \widehat{f} : \widehat{X} \to \widehat{F}, \mathsf{fill} : V(\widehat{F}) \to \{\mathsf{cur}, \mathsf{pbl}, \mathsf{ump}\}, \mathsf{load} : V(\widehat{F}) \to [\eta])$ that will give us the following information. We would like $\widehat{F}$ to correspond to the "truncation" of $F'$, when restricted to: i) the vertices in $F'$ to which vertices in $\widehat{X}$ are mapped, and ii) the vertices in $F'$ that are associated with connected components of $G_i[\widehat{Y}]$ (via $g'$). In the above not only we will preserve the mappings of $\widehat{X}$ and $\mathcal{C}(G_i[\widehat{Y}])$, but will also maintain the paths of these vertices to the root of the tree in $F'$, containing them. As the depth of $F'$ is at most $\eta$, it will be enough to have at most $\mathcal{O}(\eta(k + \eta))$ vertices in our guess $\widehat{F}$. The function $\widehat{f}$ will correspond to $f'$ restricted to the vertices in $\widehat{X}$. Some vertices in $F'$ are filled from above (vertices in $G - V(G_i)$), and thus must remain unmapped ($\mathsf{ump}$, for short), when restricted to $G_i$. Thus, the vertices in $\widehat{F}$ (if any) of the above type, will be marked $\mathsf{ump}$ by the function $\mathsf{fill}$. The vertices of $F'$ to which vertices in $\widehat{X}$ are mapped, will be assigned $\mathsf{cur}$ (short for current) by $\mathsf{fill}$, indicating that the these vertices are already used by the current bag $\beta(p_i)$, under consideration. Finally, the remaining vertices in $\widehat{F}$ are free to be potentially used by vertices that appear strictly below, and they are marked $\mathsf{pbl}$ (short for potential below) by $\mathsf{fill}$. The function $\mathsf{load}$ will indicate the depth of the sub-tree that may be to a vertex in $\widehat{F}$. We will now discuss the properties that we would like to maintain corresponding to the function $g'$. We will maintain a tuple $\mathcal{G} = (\widehat{g} : \mathcal{C}(G[\widehat{Y}]) \to V(\widehat{F}) \cup \{\perp\}, \mathsf{ext} : \mathcal{C}(G[\widehat{Y}]) \to \{0, 1\})$, which will give us the following information: the function $\widehat{g}$ correspond to the restriction of $g'$ to the connected components containing vertices from $\widehat{Y}$, and $\mathsf{ext}$ will indicate whether we can "extend" the connected component by adding vertices to it, that appear strictly below. Finally, we will maintain the guess, $k' \in \mathbb{N}$, for the size of the solution restricted to $G_i$. We can bound the number of states in the dynamic programming table by $(k + \eta)^{O(\eta k + \eta^2)} n^{\mathcal{O}(1)}$, and we can (recursively) compute each of the table entry in time bounded by $(k + \eta)^{O(\eta k + \eta^2)} n^{\mathcal{O}(1)}$. This gives us an algorithm for the problem, running in time $(k + \eta)^{O(\eta k + \eta^2)} n^{\mathcal{O}(1)}$.

[1] For the section, we will be using a modified (but equivalent) definition of $(\eta, \mathsf{Clq})$-decomposition, which will simplify some of the technicalities and arguments.

## 3.3 Polynomial-time $\mathcal{O}(\mathsf{opt} \log^2 n)$-approximation for $(\eta, \mathsf{Clq})$-ED Deletion (Theorem 2)

Our algorithm follows the recursive scheme of [1], for designing polylogarithmic approximation algorithms, that in turn builds upon the classic technique of finding balanced separators in a graph for designing approximation algorithms [26]. Our algorithm has two crucial ingredients: (i) If the graph $G$ has a clique of size $\Omega(n)$, then it also has a clique of size $\Omega(n)$ with a neighborhood of size $\mathcal{O}(\mathsf{opt} + \eta)$. Moreover, in polynomial time, either we can find such a large clique with a small neighborhood, or conclude that $G$ has no large cliques. (ii) If $G$ has no cliques of size $n/3$ and it satisfies certain simple constraints relating $\mathsf{opt}, k$ and $n$ then $G$ has a balanced separator (see Definition 6) of size $\mathcal{O}((\mathsf{opt} + \eta) \log n)$ that can be computed efficiently. In the case where $G$ does not satisfy these constraints on $\mathsf{opt}, k$ and $n$, we will have arrived at the base case of our recursion we can obtain a straightfoward approximation.

Using these two ingredients, our algorithm proceeds as follows. We try to compute a clique of size $\Omega(n)$, say $Q$, with a neighborhood of size $\mathcal{O}(\mathsf{opt} + \eta)$. If we succeed, then we add the neighbors of $Q$ to the solution, remove $Q$ from $G$, and recurse on the smaller instance. If we fail to find $Q$, then we may conclude that $G$ has no large cliques. In this case, we compute an $\mathcal{O}(\log n)$-approximate balanced separator [26], add these separator vertices to the solution, and recursively approximate the solutions in the two smaller instances. A standard induction argument on the number of vertices gives the stated bound on the approximation ratio.

## 3.4 Approximating $(\eta, \mathsf{Clq})$-decompositions (Theorem 4)

Consider an instance $(G, \eta)$ of Clq-Elimination Distance, and let $\mathbb{D} = (X, Y, F, f, g)$ be an $(\eta, \mathsf{Clq})$-decomposition of $G$. The high level idea is to identify an efficiently computable set $Z$ of vertices such that for some $F', f', g'$, $\mathbb{D}' = (Z, V(G) \setminus Z, F', f', g')$ is a $(5\eta, \mathsf{Clq})$-decomposition.

The first step of our algorithm iteratively computes a grouping of vertices of $G$. Let $\mathcal{S}$ be the set of all maximal cliques in $G$. From Observation 17, every clique in $G$ either lies on a root-to-leaf path in the interior of $\mathbb{D}$ or in the set $V(C_{\mathsf{ext}})$ for some $C \in \mathcal{C}(G[Y])$. In particular, Observation 17 holds for every set in $\mathcal{S}$. The goal of our grouping is to repeatedly combine pairs of sets in this collection while ensuring that each set in the collection obtained at every step also satisfies the statement of Observation 17.

Specifically, we begin by computing the collection $\mathcal{S}$ of all maximal cliques in the input graph $G$ and then repeatedly do the following: as long as there is a pair of sets in the collection such that either (i) they intersect in at least $\eta + 1$ vertices or (ii) the minimum set of vertices that must be deleted to separate them is at least $\eta + 1$ or (iii) both sets have size at least $2\eta + 1$ and their union can be partitioned into a clique plus at most $\eta$ vertices, then we remove both sets from the collection and add their union to the collection. We then show that every set in the collection obtained at every step in this procedure also satisfies the statement of Observation 17. Moreover, we show that for every pair of large sets (of size at least $2\eta + 1$) that remain in the collection at the end of this iterative procedure, every vertex in their intersection must be contained in the interior of any $(\eta, \mathsf{Clq})$-decomposition of $G$. Therefore, we may safely "push" these vertices (denoted by $Z_{\mathsf{nec}}$) to the interior part of the approximate decomposition that we are constructing. Now, we consider the remaining vertices of the graph and argue that vertices that appear only in small sets (sets of size at most $2\eta$) in our grouping can be safely pushed to the interior part of our approximate $(\eta, \mathsf{Clq})$-decomposition without increasing the depth of the interior by more than a constant factor. Denote the vertices pushed to the interior in this way by $Z_{\mathsf{sml}}$. We then need to deal with vertices that appear in exactly one large set in our grouping. For this, we show that in each large set, the set of vertices that appear only in this large set can be covered by a clique plus a set of at most $\eta$ vertices. We show that pushing these at most $\eta$ vertices from each large set into the interior of our decomposition also does not blow up the depth of the interior by more than a constant factor. Once we have pushed these vertices (denoted by $Z_{\mathsf{mch}}$) to the interior, we are left with a disjoint union of cliques. We argue that every set $S \in \mathcal{S}$ can be separated from its out-neighborhood by at most $\eta$ vertices and moreover, pushing an arbitrary set of such vertices into the interior for each set $S \in \mathcal{S}$ (cumulatively denoted by $Z_{\mathsf{sep}}$) also does not increase the depth of the interior by more than $\eta$. Finally, we show that the remaining vertices in $X$ that are not explicitly pushed into the interior by one of our steps can be simply removed from the interior with the result being a $(5\eta, \mathsf{Clq})$-decomposition of $G$. That is, we prove the following lemma, where $\overset{\rightsquigarrow}{X} = Z_{\mathsf{nec}} \cup Z_{\mathsf{sml}} \cup Z_{\mathsf{mch}} \cup Z_{\mathsf{sep}}$ denotes the set of vertices that we have explicitly pushed into the interior of the new decomposition.

▶ **Lemma 18.** *If there is an $(\eta, \mathsf{Clq})$-decomposition $\mathbb{D} = (X, Y, F, f, g)$ of $G$, then there is an $(\overset{\rightsquigarrow}{\eta}, \mathsf{Clq})$-decomposition $\overset{\rightsquigarrow}{\mathbb{D}} = (\overset{\rightsquigarrow}{X}, \overset{\rightsquigarrow}{Y}, \overset{\rightsquigarrow}{F}, \overset{\rightsquigarrow}{f}, \overset{\rightsquigarrow}{g})$ of $G$ such that $\overset{\rightsquigarrow}{\eta} \leq 5\eta$.*

In order to compute the 5-approximate $(\eta, \mathsf{Clq})$-decomposition, we create an appropriate graph, call it $G_{\mathsf{full}}$, on the vertex set $\overset{\rightsquigarrow}{X}$ and show that computing the treedepth exactly (and a forest embedding) of $G_{\mathsf{full}}$ is sufficient to obtain a $(5\eta, \mathsf{Clq})$-decomposition of $G$.

▶ **Lemma 19.** *There is a polynomial-time algorithm that, given a forest embedding $(F_{\mathsf{full}}, f_{\mathsf{full}})$ of $G_{\mathsf{full}}$ with depth $\eta'$, outputs an $(\eta', \mathsf{Clq})$-decomposition of $G$.*

When we use the $2^{\mathcal{O}(\mathsf{td}(\mathsf{G}_{\mathsf{full}})\cdot\mathsf{tw}(\mathsf{G}_{\mathsf{full}}))}n^{\mathcal{O}(1)}$ algorithm of Reidl et al. [30] to compute the treedepth of $G_{\mathsf{full}}$ and an optimal forest embedding, we obtain a $(5\eta, \mathsf{Clq})$-decomposition of $G$ in time $2^{\mathcal{O}(\eta^2)}n^{\mathcal{O}(1)}$. If we use the polynomial-time algorithm of Czerwinski et al. [8] to approximate the treedepth of $G_{\mathsf{full}}$, we obtain a $(\mathcal{O}(\eta^2 \log^{3/2} \eta), \mathsf{Clq})$-decomposition of $G$ in time $2^{\mathcal{O}(\eta)}n^{\mathcal{O}(1)}$.

## 4 Discussions and conclusions

We studied the parameterized complexity of detecting a small modulator to graphs that have a constant elimination distance to cluster graphs. For this problem, we obtained a polynomial kernelization and in the process, developed a $k^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$-time fixed-parameter algorithm and a polynomial-time factor-$\mathsf{min}\{\mathcal{O}(\eta \cdot \mathsf{opt} \cdot \log^2 n), \mathsf{opt}^{\mathcal{O}(1)}\}$ approximation algorithm for this problem. Since our focus was on analyzing the kernelization complexity of this problem, we have not attempted to optimize the running time of our algorithm or the exponent of $k$ in the size-bound of the kernel. Moreover, since our focus was on the "recognition" problem for such graphs, we leave for future work the design of fixed-parameter algorithms and kernelization algorithms for other problems, when parameterized by the size of the smallest modulator of the given graph to the class of $(\eta, \mathsf{Clq})$-graphs. We remark that even parameterization by the elimination distance of the input graph to cluster graphs has not been explored.

Between $\mathcal{A}_1, \mathcal{A}_2$ and Theorem 3, for the problem of computing the eliminating distance of a graph to cluster graphs, we obtained an exact algorithm (i.e., a 1-approximation) running in time $\eta^{\mathcal{O}(\eta^2)} \cdot n^{\mathcal{O}(1)}$, a 5-approximation algorithm running in time $2^{\mathcal{O}(\eta^2)} \cdot n^{\mathcal{O}(1)}$ and an $\mathcal{O}(\eta \log^{3/2} \eta)$-approximation algorithm running in time $2^{\mathcal{O}(\eta)} \cdot n^{\mathcal{O}(1)}$. An interesting direction for future research is to identify the best possible tradeoffs between approximation factor and running time for the problem. Naturally, exploring the algorithmic utility of (small modulators to) bounded elimination distance to other graph classes remains an interesting line of research.

## References

1 Akanksha Agrawal, Daniel Lokshtanov, Pranabendu Misra, Saket Saurabh, and Meirav Zehavi. Polylogarithmic approximation algorithms for weighted-f-deletion problems. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, pages 1:1–1:15, 2018.

2 Marin Bougeret, Bart M. P. Jansen, and Ignasi Sau. Bridge-Depth Characterizes which Structural Parameterizations of Vertex Cover Admit a Polynomial Kernel. *arXiv e-prints*, 2020. `arXiv:2004.12865`.

3 Marin Bougeret and Ignasi Sau. How much does a treedepth modulator help to obtain polynomial kernels beyond sparse graphs? *Algorithmica*, 81(10):4043–4068, 2019.

4 Adam Bouland, Anuj Dawar, and Eryk Kopczynski. On tractable parameterizations of graph isomorphism. In Dimitrios M. Thilikos and Gerhard J. Woeginger, editors, *Parameterized and Exact Computation - 7th International Symposium (IPEC)*, volume 7535, pages 218–230, 2012.

5 Jannis Bulian and Anuj Dawar. Graph isomorphism parameterized by elimination distance to bounded degree. *Algorithmica*, 75(2):363–382, 2016.

6 Jannis Bulian and Anuj Dawar. Fixed-parameter tractable distances to sparse graph classes. *Algorithmica*, 79(1):139–158, 2017.

7 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer Science & Business Media, 2015.

8 Wojciech Czerwinski, Wojciech Nadara, and Marcin Pilipczuk. Improved bounds for the excluded-minor approximation of treedepth. In Michael A. Bender, Ola Svensson, and Grzegorz Herman, editors, *27th Annual European Symposium on Algorithms (ESA)*, volume 144, pages 34:1–34:13, 2019.

9 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

10 Eduard Eiben, Robert Ganian, Thekla Hamm, and O-joung Kwon. Measuring what matters: A hybrid approach to dynamic programming with treewidth. In Peter Rossmanith, Pinar Heggernes, and Joost-Pieter Katoen, editors, *44th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, volume 138, pages 42:1–42:15, 2019.

11 Eduard Eiben, Robert Ganian, and Stefan Szeider. Meta-kernelization using well-structured modulators. In Thore Husfeldt and Iyad A. Kanj, editors, *10th International Symposium on Parameterized and Exact Computation (IPEC)*, volume 43 of *LIPIcs*, pages 114–126, 2015.

12 Michael R. Fellows, Bart M. P. Jansen, and Frances A. Rosamond. Towards fully multivariate algorithmics: Parameter ecology and the deconstruction of computational complexity. *Eur. J. Comb.*, 34(3):541–566, 2013.

13 Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, M. S. Ramanujan, and Saket Saurabh. Solving $d$-sat via backdoors to small treewidth. In Piotr Indyk, editor, *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 630–641, 2015.

14 Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Planar $\mathcal{F}$-deletion: Approximation, kernelization and optimal FPT algorithms. In *53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 470–479, 2012.

15 Jakub Gajarský, Petr Hlinený, Jan Obdrzálek, Sebastian Ordyniak, Felix Reidl, Peter Rossmanith, Fernando Sánchez Villaamil, and Somnath Sikdar. Kernelization using structural parameters on sparse graph classes. *Journal of Computer and System Sciences*, 84:219–242, 2017.

16 Robert Ganian, Sebastian Ordyniak, and M. S. Ramanujan. Going beyond primal treewidth for (M)ILP. In Satinder P. Singh and Shaul Markovitch, editors, *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 815–821, 2017.

17 Robert Ganian, M. S. Ramanujan, and Stefan Szeider. Combining treewidth and backdoors for CSP. In Heribert Vollmer and Brigitte Vallée, editors, *34th Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 66, pages 36:1–36:17, 2017.

18 Archontia C. Giannopoulou, Bart M. P. Jansen, Daniel Lokshtanov, and Saket Saurabh. Uniform kernelization complexity of hitting forbidden minors. *ACM Transactions on Algorithms*, 13(3):35:1–35:35, 2017.

19 Jiong Guo, Falk Hüffner, and Rolf Niedermeier. A structural view on parameterizing problems: Distance from triviality. In Rodney G. Downey, Michael R. Fellows, and Frank K. H. A. Dehne, editors, *Parameterized and Exact Computation, First International Workshop (IWPEC)*, volume 3162, pages 162–173. Springer, 2004.

20 Eva-Maria C. Hols, Stefan Kratsch, and Astrid Pieterse. Elimination distances, blocking sets, and kernels for vertex cover. In Christophe Paul and Markus Bläser, editors, *37th International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 154, pages 36:1–36:14, 2020.

21 Ashwin Jacob, Fahad Panolan, Venkatesh Raman, and Vibha Sahlot. Structural parameterizations with modulator oblivion, 2020. `arXiv:2002.09972`.

22 Bart M. P. Jansen and Marcin Pilipczuk. Approximation and kernelization for chordal vertex deletion. *SIAM J. Discrete Math.*, 32(3):2258–2301, 2018.

23 David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis. On generating all maximal independent sets. *Information Processing Letters*, 27(3):119–123, 1988.

24 Eun Jung Kim, Alexander Langer, Christophe Paul, Felix Reidl, Peter Rossmanith, Ignasi Sau, and Somnath Sikdar. Linear kernels and single-exponential algorithms via protrusion decompositions. *ACM Transactions on Algorithms*, 12(2):21:1–21:41, 2016.

25 Ton Kloks. *Treewidth, Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science*. Springer, 1994.

26 Frank Thomson Leighton and Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM*, 46(6):787–832, 1999.

27 Alexander Lindermayr, Sebastian Siebertz, and Alexandre Vigny. Elimination distance to bounded degree on planar graphs. In *45th International Symposium on Mathematical Foundations of Computer Science, MFCS*, volume 170 of *LIPIcs*, pages 65:1–65:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

28 Kazuhisa Makino and Takeaki Uno. New algorithms for enumerating all maximal cliques. In *9th Scandinavian Workshop on Algorithm Theory (SWAT)*, pages 260–272, 2004.

29 Bruce A. Reed, Kaleigh Smith, and Adrian Vetta. Finding odd cycle transversals. *Operations Research Letters*, 32(4):299–301, 2004.

30 Felix Reidl, Peter Rossmanith, Fernando Sánchez Villaamil, and Somnath Sikdar. A faster parameterized algorithm for treedepth. In *Automata, Languages, and Programming - 41st International Colloquium (ICALP)*, pages 931–942, 2014.

31 Virginia Vassilevska Williams. Multiplying matrices faster than coppersmith-winograd. In *Proceedings of the 44th Symposium on Theory of Computing Conference, (STOC)*, pages 887–898, 2012.